Cross-Validation

Nipun Batra and teaching staff

IIT Gandhinagar

July 31, 2025

Outline

The Problem with Simple Train-Test Split

Limitations of single train-test split:

- Does not use the full dataset for training and does not test on the full dataset

The Problem with Simple Train-Test Split

Limitations of single train-test split:

- Does not use the full dataset for training and does not test on the full dataset

The Problem with Simple Train-Test Split

Limitations of single train-test split:

- Does not use the full dataset for training and does not test on the full dataset
- No way to optimize hyperparameters

The Problem with Simple Train-Test Split

Limitations of single train-test split:

- Does not use the full dataset for training and does not test on the full dataset
- No way to optimize hyperparameters

The Problem with Simple Train-Test Split

Limitations of single train-test split:

- Does not use the full dataset for training and does not test on the full dataset
- No way to optimize hyperparameters
- This simple train/test split has limitations we need to address

Pop Quiz: Train-Test Split Issues

Quick Quiz 1

What's the main problem with using only a single train-test split?

a) It's too computationally expensive

Answer: b) Different splits can give very different performance estimates!

Pop Quiz: Train-Test Split Issues

Quick Quiz 1

What's the main problem with using only a single train-test split?

a) It's too computationally expensive
b) Results depend on the particular split chosen

Answer: b) Different splits can give very different performance estimates!

Pop Quiz: Train-Test Split Issues

Quick Quiz 1

What's the main problem with using only a single train-test split?

a) It's too computationally expensive
b) Results depend on the particular split chosen
c) It requires too much data

Answer: b) Different splits can give very different performance estimates!

Answer

Answer

- Does not utilize the full dataset for training

Answer

- Does not utilize the full dataset for training
- Cannot optimize hyperparameters systematically

Answer

- Does not utilize the full dataset for training
- Cannot optimize hyperparameters systematically

Answer

- Does not utilize the full dataset for training
- Cannot optimize hyperparameters systematically
- Results depend on the particular split chosen

Answer

- Does not utilize the full dataset for training
- Cannot optimize hyperparameters systematically
- Results depend on the particular split chosen
- May not get reliable performance estimates

Over multiple iterations, use different parts of the dataset for training and testing

Over multiple iterations, use different parts of the dataset for training and testing

Over multiple iterations, use different parts of the dataset for training and testing

Typically done via different random splits of the dataset

Over multiple iterations, use different parts of the dataset for training and testing

Typically done via different random splits of the dataset

Over multiple iterations, use different parts of the dataset for training and testing

Typically done via different random splits of the dataset

Challenge: How to ensure systematic evaluation?

Over multiple iterations, use different parts of the dataset for training and testing

Typically done via different random splits of the dataset

Challenge: How to ensure systematic evaluation?

Over multiple iterations, use different parts of the dataset for training and testing

Typically done via different random splits of the dataset

Challenge: How to ensure systematic evaluation?

May not use every data point for training or testing with random splits

Over multiple iterations, use different parts of the dataset for training and testing

Typically done via different random splits of the dataset

Challenge: How to ensure systematic evaluation?

May not use every data point for training or testing with random splits

Over multiple iterations, use different parts of the dataset for training and testing

Typically done via different random splits of the dataset

Challenge: How to ensure systematic evaluation?

May not use every data point for training or testing with random splits

May be computationally expensive

- Each data point is used for testing exactly once

- Each data point is used for testing exactly once
- Each data point is used for training $(k-1)/k$ of the time

- Each data point is used for testing exactly once
- Each data point is used for training $(k-1)/k$ of the time

- Each data point is used for testing exactly once
- Each data point is used for training $(k-1)/k$ of the time
- Provides more robust performance estimates

Answer

80 data points (4 out of 5 folds = $4/5 \times 100 = 80$)

Validation set helps select the best hyperparameters

Validation set helps select the best hyperparameters

Validation set helps select the best hyperparameters

Test set remains untouched until final evaluation

Validation set helps select the best hyperparameters

Test set remains untouched until final evaluation

Validation set helps select the best hyperparameters

Test set remains untouched until final evaluation

This prevents overfitting to the test set

Each fold provides one validation score

Each fold provides one validation score

Each fold provides one validation score

Process is systematic and exhaustive

Answer

Answer

- Simple CV: Used for model evaluation only

Answer

- Simple CV: Used for model evaluation only
- Nested CV: Outer loop for model evaluation, inner loop for hyperparameter tuning

Answer

- Simple CV: Used for model evaluation only
- Nested CV: Outer loop for model evaluation, inner loop for hyperparameter tuning

Answer

- Simple CV: Used for model evaluation only
- Nested CV: Outer loop for model evaluation, inner loop for hyperparameter tuning
- Nested CV provides unbiased estimates when doing hyperparameter search

Final model is trained on entire training set

Final model is trained on entire training set

Final model is trained on entire training set

Standard deviation gives confidence in results

Answer

Answer

- Single fold results can be misleading due to data variance

Answer

- Single fold results can be misleading due to data variance
- Averaging provides more robust performance estimates

Answer

- Single fold results can be misleading due to data variance
- Averaging provides more robust performance estimates

Answer

- Single fold results can be misleading due to data variance
- Averaging provides more robust performance estimates
- Reduces impact of lucky/unlucky splits

Answer

- Single fold results can be misleading due to data variance
- Averaging provides more robust performance estimates
- Reduces impact of lucky/unlucky splits
- Standard deviation indicates reliability of the estimate

Special case where $k = n$ (number of data points)

Special case where $k = n$ (number of data points)

Special case where $k = n$ (number of data points)

Each fold uses exactly one data point for testing

Special case where $k = n$ (number of data points)

Each fold uses exactly one data point for testing

Special case where $k = n$ (number of data points)

Each fold uses exactly one data point for testing

Advantages:

- Maximum use of data for training

Special case where $k = n$ (number of data points)

Each fold uses exactly one data point for testing

Advantages:

- Maximum use of data for training
- Deterministic (no randomness)

Special case where $k = n$ (number of data points)

Each fold uses exactly one data point for testing

Advantages:

- Maximum use of data for training
- Deterministic (no randomness)

Special case where $k = n$ (number of data points)

Each fold uses exactly one data point for testing

Advantages:

- Maximum use of data for training
- Deterministic (no randomness)

Special case where $k = n$ (number of data points)

Each fold uses exactly one data point for testing

Advantages:

- Maximum use of data for training
- Deterministic (no randomness)

Disadvantages:

- Computationally expensive

Special case where $k = n$ (number of data points)

Each fold uses exactly one data point for testing

Advantages:

- Maximum use of data for training
- Deterministic (no randomness)

Disadvantages:

- Computationally expensive
- High variance in estimates

Maintains class distribution in each fold

Maintains class distribution in each fold

Maintains class distribution in each fold

Important for imbalanced datasets

Maintains class distribution in each fold

Important for imbalanced datasets

Maintains class distribution in each fold

Important for imbalanced datasets

Each fold has approximately same proportion of classes

Maintains class distribution in each fold

Important for imbalanced datasets

Each fold has approximately same proportion of classes

Maintains class distribution in each fold

Important for imbalanced datasets

Each fold has approximately same proportion of classes

Example: If dataset is 70% class A, 30% class B, each fold maintains this ratio

Maintains class distribution in each fold

Important for imbalanced datasets

Each fold has approximately same proportion of classes

Example: If dataset is 70% class A, 30% class B, each fold maintains this ratio

Maintains class distribution in each fold

Important for imbalanced datasets

Each fold has approximately same proportion of classes

Example: If dataset is 70% class A, 30% class B, each fold maintains this ratio

Reduces variance in performance estimates

Answer

Answer

- Regular CV might create folds with very few (or zero) positive examples

Answer

- Regular CV might create folds with very few (or zero) positive examples
- This would give misleading performance estimates

Answer

- Regular CV might create folds with very few (or zero) positive examples
- This would give misleading performance estimates

Answer

- Regular CV might create folds with very few (or zero) positive examples
- This would give misleading performance estimates
- Stratified CV ensures each fold has $\sim$10% positive examples

Answer

- Regular CV might create folds with very few (or zero) positive examples
- This would give misleading performance estimates
- Stratified CV ensures each fold has $\sim$10% positive examples
- Results in more reliable and consistent evaluation

Regular CV assumes data points are independent

Regular CV assumes data points are independent

Regular CV assumes data points are independent

Time series data has temporal dependencies

Regular CV assumes data points are independent

Time series data has temporal dependencies

- Regular CV assumes data points are independent
- Time series data has temporal dependencies
- **Forward Chaining:** Train on past, test on future

Regular CV assumes data points are independent

Time series data has temporal dependencies

Forward Chaining: Train on past, test on future

Regular CV assumes data points are independent

Time series data has temporal dependencies

Forward Chaining: Train on past, test on future

Rolling Window: Fixed-size training window

Regular CV assumes data points are independent

Time series data has temporal dependencies

Forward Chaining: Train on past, test on future

Rolling Window: Fixed-size training window

Regular CV assumes data points are independent

Time series data has temporal dependencies

Forward Chaining: Train on past, test on future

Rolling Window: Fixed-size training window

Expanding Window: Growing training set over time

Regular CV assumes data points are independent

Time series data has temporal dependencies

Forward Chaining: Train on past, test on future

Rolling Window: Fixed-size training window

Expanding Window: Growing training set over time

Regular CV assumes data points are independent

Time series data has temporal dependencies

Forward Chaining: Train on past, test on future

Rolling Window: Fixed-size training window

Expanding Window: Growing training set over time

Never use future data to predict past!

Data Leakage: Information from test set influences training

Data Leakage: Information from test set influences training

Data Leakage: Information from test set influences training

Incorrect Splitting: Not accounting for grouped data

Data Leakage: Information from test set influences training

Incorrect Splitting: Not accounting for grouped data

Data Leakage: Information from test set influences training

Incorrect Splitting: Not accounting for grouped data

Overfitting to CV: Too much hyperparameter tuning

Data Leakage: Information from test set influences training

Incorrect Splitting: Not accounting for grouped data

Overfitting to CV: Too much hyperparameter tuning

Data Leakage: Information from test set influences training

Incorrect Splitting: Not accounting for grouped data

Overfitting to CV: Too much hyperparameter tuning

Wrong Preprocessing: Scaling on entire dataset before splitting

Data Leakage: Information from test set influences training

Incorrect Splitting: Not accounting for grouped data

Overfitting to CV: Too much hyperparameter tuning

Wrong Preprocessing: Scaling on entire dataset before splitting

Data Leakage: Information from test set influences training

Incorrect Splitting: Not accounting for grouped data

Overfitting to CV: Too much hyperparameter tuning

Wrong Preprocessing: Scaling on entire dataset before splitting

Ignoring Class Imbalance: Not using stratified CV when needed

Answer

Answer

- This causes data leakage!

Answer

- This causes data leakage!
- Test fold statistics influence the training preprocessing

Answer

- This causes data leakage!
- Test fold statistics influence the training preprocessing

Answer

- This causes data leakage!
- Test fold statistics influence the training preprocessing
- Should compute statistics only on training folds

Answer

- This causes data leakage!
- Test fold statistics influence the training preprocessing
- Should compute statistics only on training folds
- Apply same transformation to corresponding test fold

Answer

- This causes data leakage!
- Test fold statistics influence the training preprocessing
- Should compute statistics only on training folds
- Apply same transformation to corresponding test fold
- This gives more realistic performance estimates

Better Data Utilization: Every point used for both training and testing

Better Data Utilization: Every point used for both training and testing

Better Data Utilization: Every point used for both training and testing

Robust Evaluation: Multiple train/test splits reduce variance

Better Data Utilization: Every point used for both training and testing

Robust Evaluation: Multiple train/test splits reduce variance

Better Data Utilization: Every point used for both training and testing

Robust Evaluation: Multiple train/test splits reduce variance

Hyperparameter Tuning: Systematic way to select best parameters

Better Data Utilization: Every point used for both training and testing

Robust Evaluation: Multiple train/test splits reduce variance

Hyperparameter Tuning: Systematic way to select best parameters

Better Data Utilization: Every point used for both training and testing

Robust Evaluation: Multiple train/test splits reduce variance

Hyperparameter Tuning: Systematic way to select best parameters

Model Comparison: Fair comparison between different algorithms

Better Data Utilization: Every point used for both training and testing

Robust Evaluation: Multiple train/test splits reduce variance

Hyperparameter Tuning: Systematic way to select best parameters

Model Comparison: Fair comparison between different algorithms

Better Data Utilization: Every point used for both training and testing

Robust Evaluation: Multiple train/test splits reduce variance

Hyperparameter Tuning: Systematic way to select best parameters

Model Comparison: Fair comparison between different algorithms

Confidence Estimates: Standard deviation indicates reliability

K-Fold (k=5,10): General purpose, most common

K-Fold (k=5,10): General purpose, most common

K-Fold (k=5,10): General purpose, most common

Stratified: Imbalanced classification problems

K-Fold (k=5,10): General purpose, most common

Stratified: Imbalanced classification problems

K-Fold (k=5,10): General purpose, most common

Stratified: Imbalanced classification problems

LOOCV: Small datasets, when computational cost is acceptable

K-Fold (k=5,10): General purpose, most common

Stratified: Imbalanced classification problems

LOOCV: Small datasets, when computational cost is acceptable

K-Fold (k=5,10): General purpose, most common

Stratified: Imbalanced classification problems

LOOCV: Small datasets, when computational cost is acceptable

Time Series CV: Temporal data with dependencies

K-Fold (k=5,10): General purpose, most common

Stratified: Imbalanced classification problems

LOOCV: Small datasets, when computational cost is acceptable

Time Series CV: Temporal data with dependencies

K-Fold (k=5,10): General purpose, most common

Stratified: Imbalanced classification problems

LOOCV: Small datasets, when computational cost is acceptable

Time Series CV: Temporal data with dependencies

Nested CV: When doing extensive hyperparameter search

Always preprocess within each fold separately

Always preprocess within each fold separately

Always preprocess within each fold separately

Use stratification for classification problems

Always preprocess within each fold separately

Use stratification for classification problems

Always preprocess within each fold separately

Use stratification for classification problems

Report mean $\pm$ standard deviation

Always preprocess within each fold separately

Use stratification for classification problems

Report mean $\pm$ standard deviation

Always preprocess within each fold separately

Use stratification for classification problems

Report mean $\pm$ standard deviation

Don't overfit to cross-validation results

Always preprocess within each fold separately

Use stratification for classification problems

Report mean $\pm$ standard deviation

Don't overfit to cross-validation results

Always preprocess within each fold separately

Use stratification for classification problems

Report mean $\pm$ standard deviation

Don't overfit to cross-validation results

Consider computational cost vs. benefit trade-off

Always preprocess within each fold separately

Use stratification for classification problems

Report mean $\pm$ standard deviation

Don't overfit to cross-validation results

Consider computational cost vs. benefit trade-off

Always preprocess within each fold separately

Use stratification for classification problems

Report mean $\pm$ standard deviation

Don't overfit to cross-validation results

Consider computational cost vs. benefit trade-off

Use nested CV for unbiased hyperparameter search

Next time: Ensemble Learning

- How to combine various models?

Next time: Ensemble Learning

- How to combine various models?
- Why combine multiple models?

Next time: Ensemble Learning

- How to combine various models?
- Why combine multiple models?

Next time: Ensemble Learning

- How to combine various models?
- Why combine multiple models?
- How can we reduce bias?

Next time: Ensemble Learning

- How to combine various models?
- Why combine multiple models?
- How can we reduce bias?
- How can we reduce variance?

Next time: Ensemble Learning

- How to combine various models?
- Why combine multiple models?
- How can we reduce bias?
- How can we reduce variance?
- Bootstrap aggregating (Bagging)

Next time: Ensemble Learning

- How to combine various models?
- Why combine multiple models?
- How can we reduce bias?
- How can we reduce variance?
- Bootstrap aggregating (Bagging)

Next time: Ensemble Learning

- How to combine various models?
- Why combine multiple models?
- How can we reduce bias?
- How can we reduce variance?
- Bootstrap aggregating (Bagging)
- Boosting methods